

Checking the boundaries of static analysis

Latest slides: <http://goo.gl/pcDkV>

Introduction

Lots of interesting research

In academia, two big different flavors:

Abstract Interpretation

SMT-centric analysis

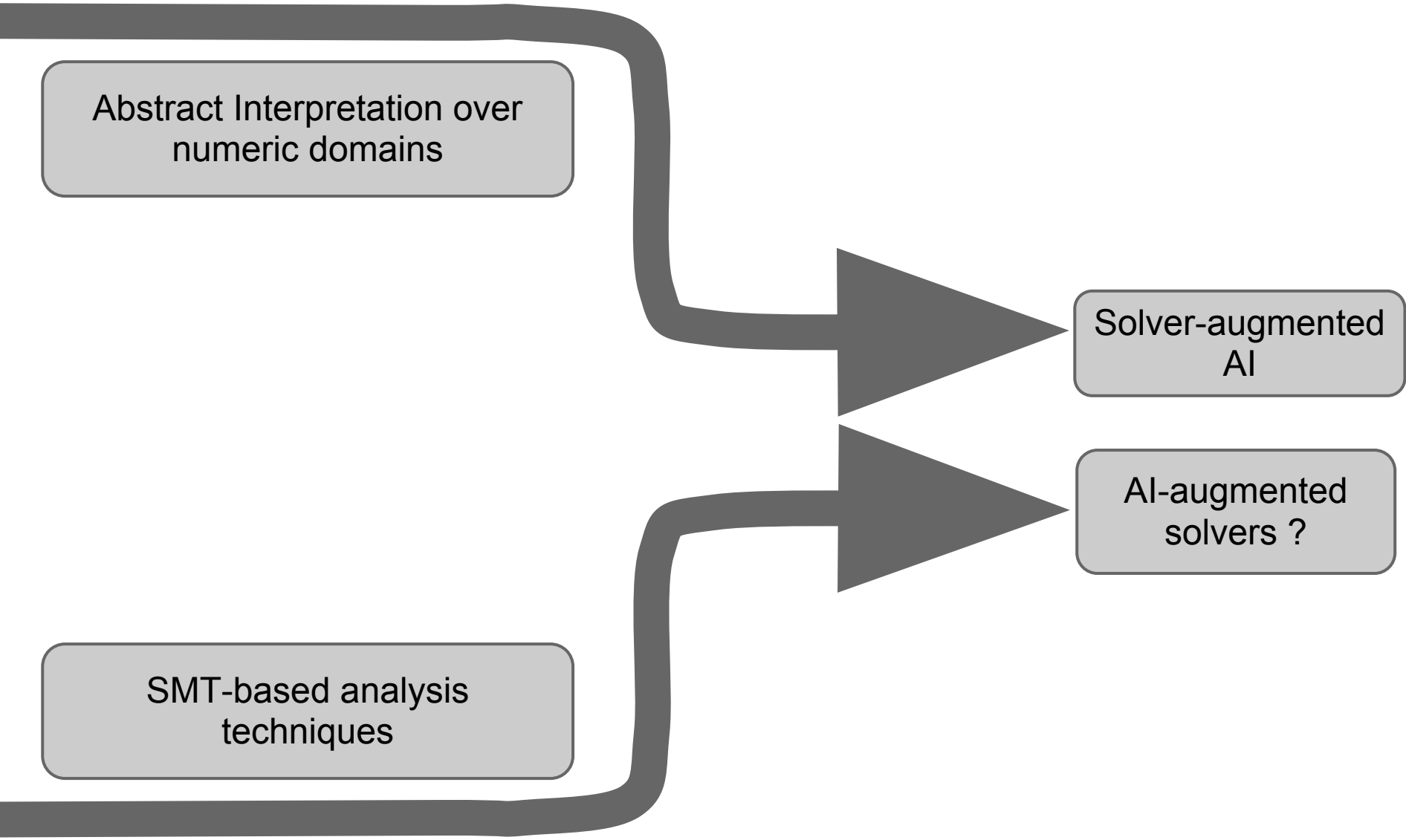
Convergence of the approaches

Abstract Interpretation over
numeric domains

SMT-based analysis
techniques

Solver-augmented
AI

AI-augmented
solvers ?



Goal of the talk (Part 1)

Vague introduction to the two flavors

Common weaknesses of AI implementations

Possible improvements through application of
SMT solvers

Goal of the talk (Part 2)

What bugs are hard to find ?

What bugs are easy to find ?

Why are browsers the "perfect storm" for static
analyzers ?

Important

Survey talk

"Other people's work"

Extensive bibliography at the end

SMT-based analyses

Idea: Generate set of equations from code

Common form: boolean variables,
"bitvectors" and "arrays"

SMT-solver finds one satisfying variable
assignment or UNSAT

Abstract Interpretation

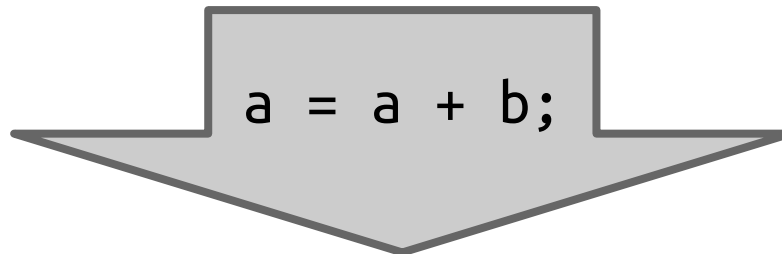
"In line X of the program, the property Y will always be true."

Derives such statements automatically in a structured way

Deal with questions of decidability / halting / tractability

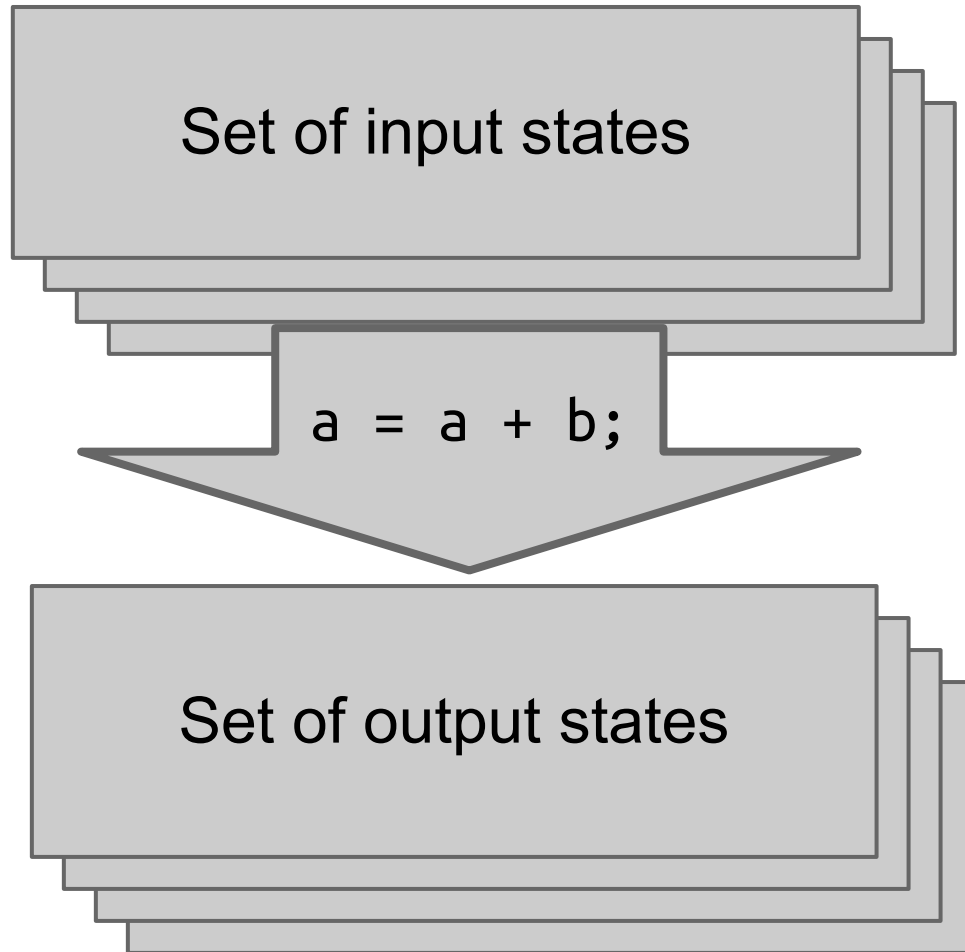
Concrete Execution

Input state:
 $a = 4, b = 10$

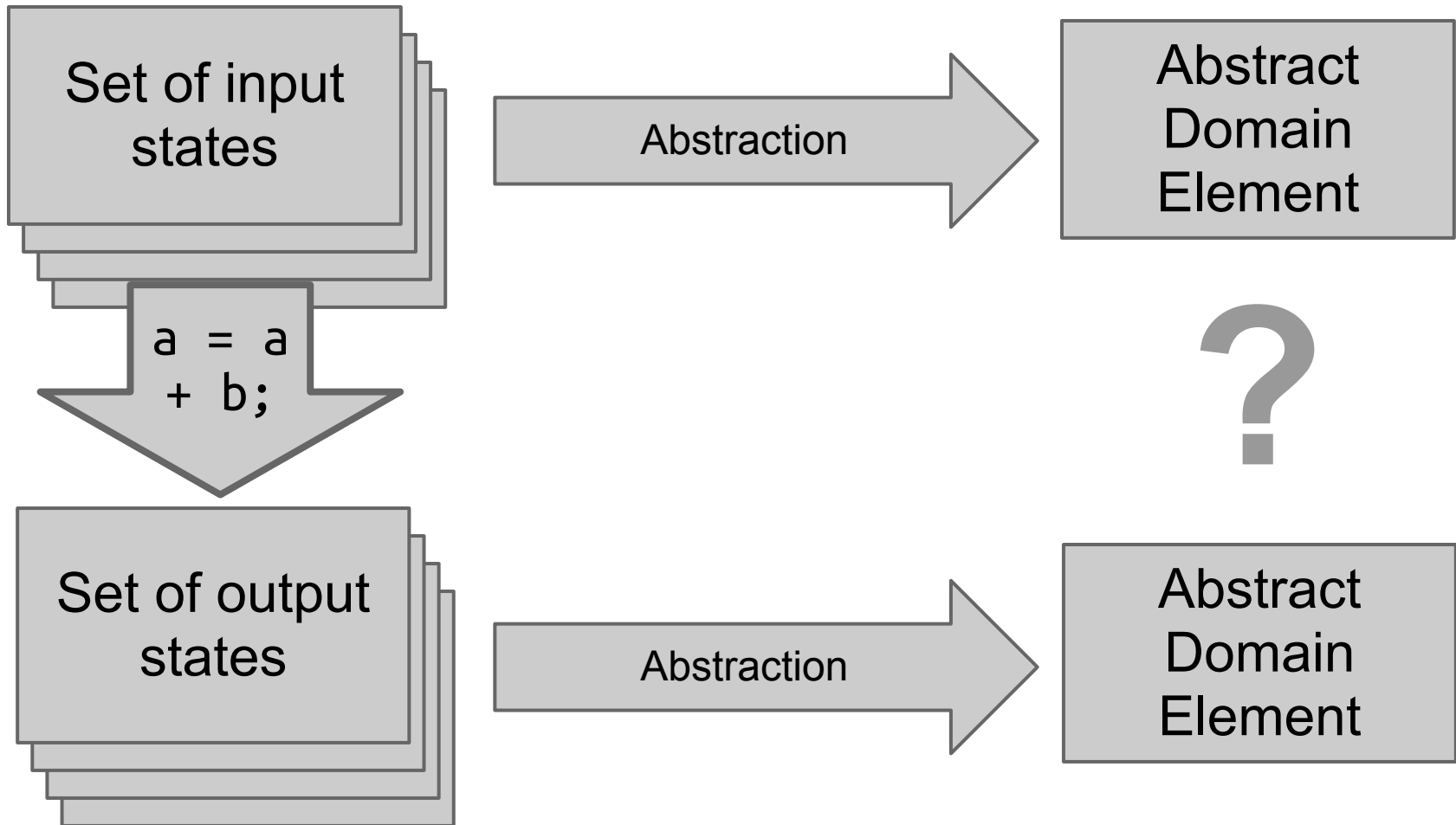


Output state:
 $a = 14, b = 10$

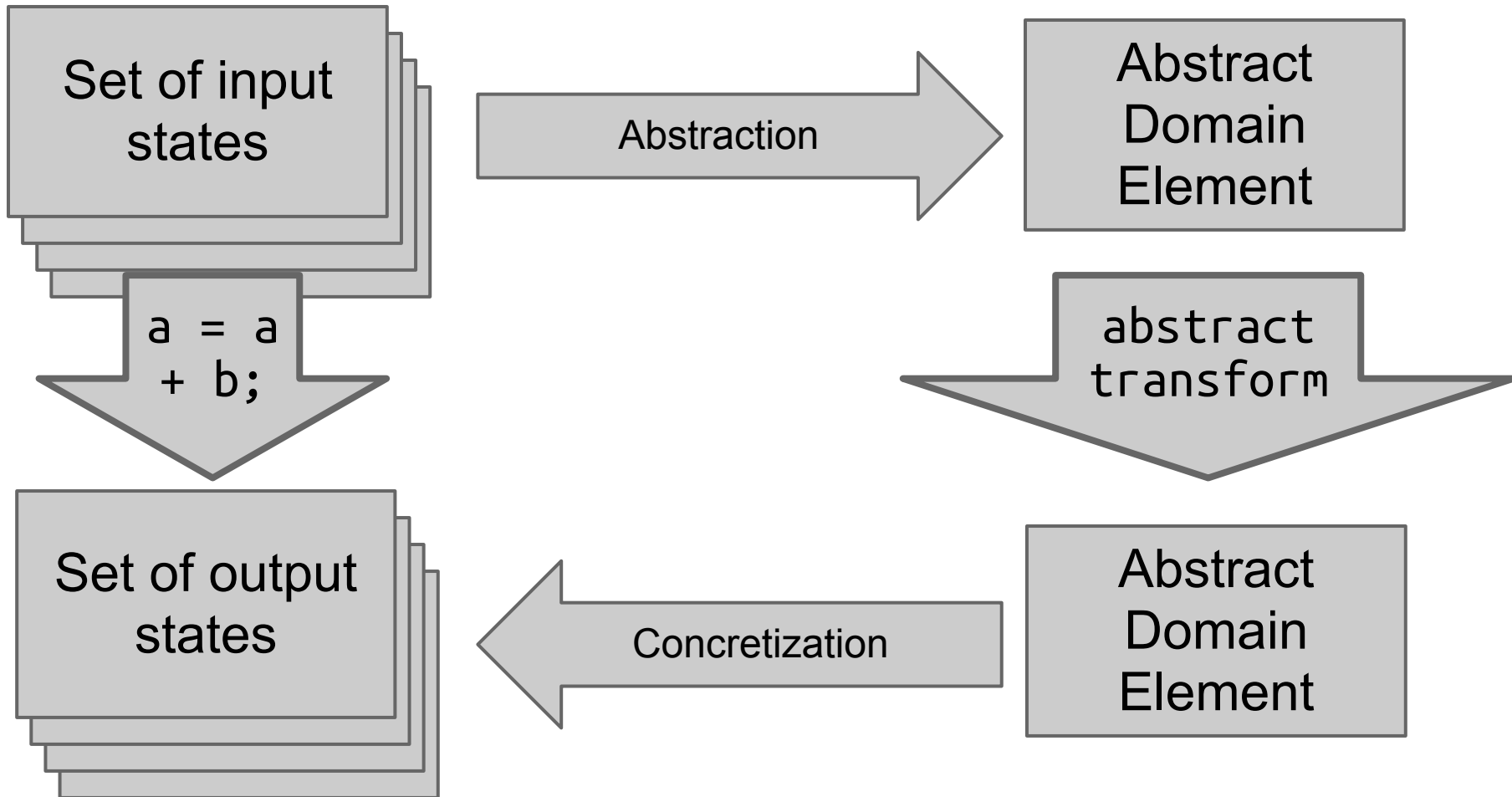
Concrete Execution



Concrete vs. Abstract Interpretation



Concrete vs. Abstract Interpretation



Abstract vs. concrete domains

Compute on "simpler" structures.

Need "join" ("set union") and "intersect".

The effect of the code line needs to be translated, too.

Example: Intervals

Incoming set of states contains 990 elements

$$\left\{ \begin{array}{l} (a = 20, b=5), \\ (a = 22, b=5), \\ \dots, \\ (a = 2000, b = 5) \end{array} \right\}$$

Approximate with *interval*:

(a is in [20...2000], b is in [5...5])

Intervals can be joined and intersected.

Compute on abstract domain

$a \text{ in } [20..2000], b \text{ in } [5..6]$

$a = a + b;$

$a \text{ in } [25..2006], b \text{ in } [5..6]$

"Abstract Interpretation"

Interpreting (~executing) the program on an abstract domain instead of on a concrete set of examples.

Central Questions

Properties of good abstract domains ?

What abstract domains can we think of ?
Intervals, polyhedral domains etc.

How to lift concrete operations to abstract domains ?

Common pitfalls & problems

Codependency of intermediate representation
and abstract domains

Clumsyness of constructing products of
domains

Myopic static analysis

Summarization of heap cells and implications

Improve with solvers ?

Codependency of intermediate representation
and abstract domains

Clumsyness of constructing products of
domains

Myopic static analysis

Summarization of heap cells and implications

Myopic static analysis

Sequence of instructions

Each instruction is lifted to abstract domain

Each instruction overapproximates concrete
operation

Cascade of imprecision leads to failure

Myopic static analysis

$$op_1 \circ op_2 \circ \dots \circ op_n$$

$$\alpha(op_1) \circ \alpha(op_2) \circ \dots \circ \alpha(op_n)$$

$\neq$

$$\alpha(op_1 \circ op_2 \circ \dots \circ op_n)$$

Myopic static analysis

```
x1 = x & 0xFF
```

```
x2 = x & 0xFF00
```

```
x3 = x & 0xFF0000
```

```
x4 = x & 0xFF000000
```

```
res = x1
```

```
res = res | x2
```

```
res = res | x3
```

```
res = res | x4
```